UNIT-2

Formal relational Query Languages.

The required tables are as follows for this entire chapter.

1. Instructor relation.

ID	NAME	DEPT-NAME	salary
10101	Srinivasan	CSE	65,000/-
12121	Who	FINANCE	90,000/-
15151	MORZET	MUSIC	40,000/-
22222	Finstein	physics	95,000/-
32343	Alisyed	History	80,000/-
33456	gold	physics	87,000/-
45561	kaza	CSE	75,000/-
58583	kali	history	62,000/-
76543	singh	finance	80,000/-
76766	krick	Biology	72,000/-
83821	BrandT	CSE	92,000/-
98345	Kim	EEE	80,000/-

2. Section

Course ID	SEC ID	semester	Year	building	Room no.	Time ID
BIO-101	1	Summer	2009	SRK	514	B
BIO-301	1	Summer	2010	SRK	514	A
CS-101	1	Fall	2009	BG	101	H
CS-101	1	Spring	2010	BG	101	F
CS-110	1	Spring	2009	CB	128	E
CS-190	2	Spring	2009	CB	128	A

Scanned by CamScanner

CS-360	1	spring	2010	MC	120	D
CS-319	1	spring	2010	MC	100	B
CS-319	2	spring	2010	ZB	128	C
CS-347	1	Fall	2009	CB	120	H
EE-181	1	spring	2009	CB	128	C
FIN-201	1	spring	2010	BG	101	B
HIS-351	1	spring	2010	SPK	514	C
MU-199	1	spring	2010	SPK	101	D
PHY-101	1	fall	2009	MC	100	A

3. teaches.

ID	courseID	secID	semester	year
10101	CS-101	1	fall	2009
10001	CS-315	1	spring	2010
10101	CS-347	1	fall	2009
12121	FIN-201	1	spring	2010
15151	MU-199	1	spring	2010
22222	PHY-101	1	fall	2009
32343	HIS-351	1	spring	2010
45565	CS-101	1	Spring	2010
45566	CS-319	1	Spring	2010
76766	BIO-101	1	Summer	2009
76766	BIO-301	1	summer	2010
83821	CS-190	1	spring	2009
83821	CS-190	2	spring	2009
83821	CS-319	2	Spring	2010
98345	EE-181	1	Spring	2009

Scanned by CamScanner

Relational algebra:-

The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation as a result.

The fundamental operations in relational algebra are
1. select
2. project
3. Union
4. set difference
5. cartesian product
6. Rename

They are several other operations in the relational Algebra namely.
1. set intersection
2. natural join
3. assignment.

→ select, projection, union, rename operations are called unary operations. remaining all are binary operations.

The select operation:

The select operation selects tuple that satisfy a given predicate. we use the lower greek letter (σ) sigma, to denote selection.

The term select in relational algebra has a different meaning than the one used in SQL.

Scanned by CamScanner

which is an unfortunate historical fact.

→ In relational algebra the term select corresponding to what we refer to in SQL as ~~where~~.

where

Select ename, sal from emp where sal > 90000;

→ In the relation, instructor which one is the belongs to physics dept (or) Details of instructors which belongs to physics.

$\sigma_{Dept_name = 'physics'}(Instructor)$

→ all the instructors ~~which~~ salary is > 90000

$\sigma_{sal > 90000}$ (~~emp~~) (instructor)

→ ~~which~~ is ~~the~~ all the persons which draw greater than 90000 in the physics department.

and = ∧
OR = ∨
NOT = ¬.

$\sigma_{sal > 90000 \wedge Dept_physics}$ (Instructor).

Projection operation:-

Suppose we want to list all the instructor's id and name only then the project operation allows us to produce this result.

Projection is denoted by the uppercase greek letter (Π)

$\Pi_{col_1, col_2, \ldots}$ (Relation name)

Display all the instructors id name from instructor table.

Scanned by CamScanner

$\Pi_{ID, name}$ (Instructor).

Composition of relational operations :-

Consider the more complicated Query like find the names of all the instructors in the physics department

Π_{name} ($\sigma_{dept_name = physics}$ (Instructor))

The Union operation :-

To find the set of all courses taught in the fall 2009 semester or courses taught in spring 2010 semester or both.

Π_{course_ID} ($\sigma_{semister = 'fall' \wedge year = 2009}$ (section)) $\cup$

Π_{course_ID} ($\sigma_{semister = 'spring' \wedge year = 2010}$ (section)).

course-ID

CS-101	FIN-201
CS-101	HIS-351
CS-350	MAY-101.
CS-319	
CS-347.	

The set difference operation :-

The set difference operation $-$ denoted by $(-)$ allows us to find the tuples that are in one relation but are not in another relation.

The expression r-s produces a relation containing those tuples in r but not in s.

Scanned by CamScanner

where $r-s \neq s-r$.

→ find all the courses in the fall 2009 semester but not in spring 2010 semester.

$$\Pi_{courseid}(\sigma_{semester='fall' \wedge year=2009}(section)) - \Pi_{courseid}(\sigma_{semester='spring' \wedge year=2010}(section))$$

Cartesian product operation:

It is denoted by 'x' allows us to combine any two relations we write the cartesian relation of r & s as r x s

$$\Pi_{name, course\text{-}ID}(\sigma_{instructor.ID = teaches.ID}(\sigma_{dept_name = physics}(instructor \times teaches)))$$

Result

name	course-ID
Einstein	phy-101

Instructor id is in the instructor id it does not need to teach any course.

∴ It does not appear in result.

→ The rename operator is denoted by lowercase greek letter (ρ). rho.

$\rho_{newname}(oldname)$.

new name and old names are the relation names.

Scanned by CamScanner

Formal definition of relational algebra:

A basic expression in relational algebra contains either one of the following

i) a relation in the database

ii) a constant in a relation.

A constant relation is written by listing its tuples with in parenthesis ({ . }) for example

{22222, Einstein, physics, 95000}

Let E_1, E_2 be relational algebraic expressions then following all are ~~algebraic~~ relational algebraic expressions

Eg: 1) $E_1 \cup E_2$

2) $E_1 - E_2$

3) $E_1 \times E_2$

4) $\sigma_p(E_1)$ where 'p' is a predicate on attributes in E_1.

$\Pi_s(E_1)$ where 's' is a list consisting of some of the attributes in E_1.

$\rho_x(E_1)$ where 'x' is the new name for the result of E_1.

additional relational algebra expressions:-

→ the set intersection operation:

The first additional relational operation that we shall define is intersection ($\cap$).

Scanned by CamScanner

Suppose that we wish to find all courses taught in both the fall 2009 and the spring 2010 semester. (~~section~~)

$\Pi_{course\text{-}ID}(\sigma_{semester='fall' \wedge year=2009}(section)) \cap \Pi_{course\text{-}ID}(\sigma_{semester='spring' \wedge year=2010}(section))$.

Note that we can rewrite any relational algebra expression that uses intersection by replacing the intersection operation with a pair of set difference operations as $r \cap s = r - (r - s)$

→ The Natural join operation:-

The natural join is a binary operation that allows us to combine certain selections and a cartesian product into one operation. It is denoted by the join symbol "$\bowtie$".

→ The natural join operation forms a cartesian product of its two arguments, performs a selection ~~course~~ forcing equality on those attributes that appears in the both relational schemas and finally removes duplicate attributes.

Eg: (Instructor $\bowtie$ teaches)

(ID, Name, Dept_name) (~~name~~, ~~Dept_D~~

(course ID, sec ID, semester, year)

10101 Srinivasan CSE 65000 CS-101 1 fall 2009

10101 Srinivasan CSE 65000 CS-350 1 spring 2010.

10101 Srinivasan CSE 65000 CS347 1 fall 2009

Scanned by CamScanner

12121 who finance 90000 FIN-201

Assignment operation :-

The assignment operation denoted by (←) is assigned to the relation variable on the left of the arrow to the right of the arrow

$temp_1 \leftarrow R \times S$

$temp_2 \leftarrow \Pi_{ID, name}(Instructor)$.

Outer join operation:

There are 3 types of outer join operation

1. left outer join
2. right outer join
3. full outer join.

Left outer join :- (⟕)

Denoted by (⟕) takes all the tuples in the left relation that didn't match with any tuple in the right relation. pads the tuples with null value for all other attributes and adds them to the result of the natural join

Eg: (33456, Gold, physics, 87000, NULL, NULL NULL, NULL)

Scanned by CamScanner

Right outer join:- (⟖)

It is a symmetric with left outer join it pads tupples from the right relation that didnot match any from the left relation with NULLs. and adds them to the result of the natural join.

Eg: (teaches ⟖ instructor)

(78583, NULL, NULL, NULL, NULL, califery, his, 62000)

Full outer join:- (⟗)

Thus both the left and right outer join operation pading tupples from left relation that didnot match any from the left relation and adding these tupples to the natural join.

Extended relational algebra operations:-

generalised projection:-

find the id, name, yearly calary of all the instructors

$\Pi_{ID, name, salary * 12}$ (Instructor).

Aggregation:-

Aggregation function takes a collection of values and return a single value as a result. The following are the aggregate function.

1. COUNT
2. SUM
3. Average
4. distinct

Scanned by CamScanner

Queries

⇒ find the average salary of the all instructors.

$$\mathcal{G}_{Avg(salary)}(Instructor)$$

⇒ find the sum of all the salary

$$\mathcal{G}_{sum(salary)}(Instructor)$$

Tupple relational calculus :-

When we write the relational algebra expression we follow sequence of procedure that generates the answer to the Query.

The tuple relational calculus is a non procedural query language. It describes information without giving a specific procedure for obtaining that information.

A query in the tuple relational calculus is expressed as $\{t / P(t)\}$ i.e it is a set of all tuples t/ predicate p is True for t. and also we use $t[A]$ to denote the value of tuple t on attribute A. and we use $t \in r$ to denote the tuple t is in relation r.

After some examples we give a formal definition of the tuple relational calculus.

Eg: find the id, name, dept_name, salary for the instructors whose salary is > 80000.

Scanned by CamScanner

$\{ t / t \in \text{Instructor} \wedge t[\text{salary}] > 80000 \}$

Suppose that we want only ID attribute rather than all the attributes of the instructor.

to write this query in TRC we need to write an expression for a relation on schema (id) we need those tuples ID such that there is a tuple instructor with the salary attribute is >80000.

TO express this request we need the exists ($\exists$)

There exists a tuple t in relation r such that the predicate $Q(t)$ is true.

$$\exists t \in r (Q(t))$$

Query for only ID:

$$\{ t / \exists s \in \text{Instructor} (t[\text{ID}] = s[\text{ID}] \wedge s[\text{salary}] > 80000) \}$$

Consider the Query to find all the students who have taken all courses offered in biology department

TO write this Query ~~TRL~~ we use ($\forall$) construct denoted by (forall)

$\forall t \in r (Q(t))$ that means Q is true for all tuples t in relation r.

$\{ t / \exists r \in \text{student} (r[\text{ID}] = t[\text{ID}] \wedge (\forall u \in \text{course} (u[\text{dept-name}] = \text{'Biology'} \Rightarrow \exists s \in \text{takes} ($

Scanned by CamScanner

$t(ID = s[ID] \wedge s[course_ID] = u[course_ID])$

Formal definition:-

The TRC expression is of the form $\{t / P(t)\}$ where P is a formula several tuple variable may appear in a formulae.

A tuple variable is said to be free variable unless it is quantified by a $\exists$ (or) $\forall$

Eg: ~~free~~ 't' belongs to instructor and 's'

$t \in instructor \wedge \exists s \in dept$ ($t[dept_name] = s[dept_name]$)

Here t is a free variable and s is said to be a bounded variable.

The tuple relational calculus formulae is built up out of atoms. An atom has one of the following forms

1. $s \in r$, where s is a tuple variable and r is a relation. [we do not allow use of the $\notin$ operator]
2. $s[x] \Theta u[y]$ where s, and u are tuple variables, x is an attribute on which s is defined. y is an attribute on which u is defined.

Θ is comparison operator ($<, >, >=, <=, =, \neq$).

Scanned by CamScanner

we require the attribute x & y have domain's whose members can be compared by Θ.

3. $s[x] \Theta c$.

where 's' is a tuple variable, x is an attribute on which s is defined and Θ is a comparison operator, and c is a constant in the domain of attribute x.

We build up formulae from atoms by using the following rules.

1) An atom is a formulae. if P_1 is a formulae
2) than so $\neg P_1$ and (P_1) are also formula.
3) if P_1 and P_2 are formula then P_1 and P_2 $P_1 \wedge (P_1 \wedge P_2)$, $P_1 \vee P_2$, $P_1 \Rightarrow P_2$ are formulae
4) if $P_1(s)$ is formulae containing a free tuple variable s and r E is a relation than $\exists s \in r(P_1(s))$ and $\forall s \in r(P_1(s))$ are also formulae.

There are 3 equalences

1. P_1 and P_2 is $P_1 \wedge P_2 = \neg(\neg(P_1) \vee \neg(P_2))$

$\forall t \in r(P_1(t)) = \neg \exists t \in r(\neg P_1(t))$

$P_1 \Rightarrow P_2 = \neg(P_1) \vee P_2$.

Scanned by CamScanner

Safety of expressions:-

There are 1 final issue to be discussed. A tuple relational calculus expression may generate an infinite relation. Suppose that

$\{t / \neg(t \in \text{instructor})\}$

There are infinitely many tuples that are not in instructor. most of these tuples contain values that do not even appear in the database. To help define a relation of tuple relational calculus we introduce the concept of domain of a tuple relational calculus.

The domain of P is denoted DOM(P). It is the set of all values referenced by P.

The domain relational calculus:-

The second form of relational calculus called as domain relational calculus uses domain variables that takeon variable from an attribute domain, rather than values for an entire tuple.

The domain relational calculus, however is closely related to the tuple relational calculus. The domain relational calculus serves as theoritical basis for the widly used QBE (Query by Example) language.

Scanned by CamScanner

Formal definition:

—An Expression in domain relational calculus is of the form $\{\langle x_1, x_2, \ldots x_n \rangle / P(x_1, x_2, \ldots x_n)\}$. where $x_1, x_2, \ldots x_n$ represents the domain variables, P represents the formulae composed of atoms.

An atom in the domain relational calculus has one of the following forms:

$\langle x_1, x_2, \ldots x_n \rangle \in r$ where r is a relation on n attributes. and $x_1, x_2, \ldots x_n$ are domain variables. (or) domain constants.

$x \Theta y$ where x and y are domain variables. and Θ is a comparison operator ($<, >, <=, >=, =, \neq$). We require that attributes x and y have domains that can be compared by Θ.

$x \Theta c$ where x is the domain variable Θ is a comparison operator and c is a constant in the domain of the attribute for which x is a domain variable

We buildup formulae from atoms by using the following rules:

1. An atom is a formulae.
2. if P_1 is a formulae then $\neg P_1$ and (P_1) are also formulae.

Scanned by CamScanner

3) if P_1 and P_2 are formulae then $P_1 \vee P_2$, $P_1 \wedge P_2$, $P_1 \Rightarrow P_2$ are also formulae.

4) if $P_1(x)$ is a formulae in 'x' where 'x' is a free domain variable then there exists $\exists P_1(x)$ and $\forall P_1(x)$ are also formulae.

Example Queries:-

1. find the instructor id, name, dept_name and salary for the instructor whose salary > 80000

$\{ \langle i, n, d, s \rangle / \langle i, n, d, s \rangle \in \text{Instructor} \wedge s > 80000 \}$

2. find the all instructor id's for instructor whose salary > 80000.

$\{ \langle i \rangle / \exists n, d, s (\langle i, n, d, s \rangle \in \text{Instructor} \wedge s > 80000) \}$

3. find the names of all instructors in physics dept together with the course id of all the courses they teach.

$\{ \langle n, c \rangle / \exists i, a (\langle i, c, a, s, y \rangle \in \text{teaches} \wedge \exists d, s (\langle i, n, d, s \rangle \in \text{instructor} \wedge d = \text{'physics'}) \}$.

Scanned by CamScanner

Safety of ~~relation~~ expression DRC:

We noted that in the tuple relational calculus it is possible to write expressions that may generate an infinite relation that leads us to safety for tuple relational calculus expression. A similar situation occurs for the domain relational calculus : an expression such as

$\{ \langle i, n, d, s \rangle \mid \neg (\langle i, n, d, s \rangle \in \text{Instructor}) \}$.

$\{ \langle x_1, x_2 \ldots x_n \rangle \mid \neg P(x_1, x_2, \ldots x_n) \}$ is safe if all the following hold:

i) All values that appear in tuples of the expression are values from dom(P).

ii) for every ~~their~~ exists subformulae of the form $\exists x P(x)$, the subformulae is true if and only if there exists a value of x in dom(P) such that $P_1(x)$ is true.

iii) for every forall subformulae of the form $\forall x (P(x))$, the subformulae is ~~to~~ if and only if $P(x)$ is true for all values x from dom(P).

Scanned by CamScanner

Introduction to SQL:

Overview of SQL Query language:

First IBM developed the original version of SQL, originally called "Sequel" as part of the System 'R' project in the early 1970's.

The Sequel language has evolved since then and its name has changed to 'SQL'. (Structured Query Language). Many products now support the SQL language. In 1986 the American National Standard Institute (ANSI) and the International Standard Organisation for standard organization (ISO) published an SQL language called SQL-86, ANSI published an extended standard for SQL, SQL-89 in 1989.

The Next Versions of the standards for SQL-92 followed by 1999, SQL-2003, SQL-2006 and most recently SQL-2008.

The SQL language has several parts

1. Data definition language.
2. Data manipulation language.

Integrity:

The SQL DDL includes commands for specify Integrity constraints that the data stored

Scanned by CamScanner

in the database must satisfy. updates that violates integrity constraints are disallowed.

View definition:

The SQL DDL includes commands for defining views.

Transaction control:

SQL includes commands for specifing the beginning and ending of transactions.

Embeded SQL and Dynamic SQL:

it defines how SQL statements can be embedded with in general purpose programming languages such as C, C++, java.

Authorisation:

The SQL DDL includes commands for specifying access ~~rights~~ rights to relations view.

SQL data definition:

The set of relations in the database must be specified to the system by means of a data definition language. The SQL ddl allows specification of not only a set of relations but also information about each relation including

Scanned by CamScanner

1. The schema for each relation.
2. The type of values associated with each attribute
3. The integrity constraints.
4. The set of indices to be maintained for each relation.
5. The security and authorization information for each relation
6. The physical storage structure of each location ~~relation~~ on ~~the~~ disc.

The basic datatypes :-

1. char(n)
2. varchar(n).
3. int
4. Small int
5. numeric (P, d)
6. real
7. float(n).

Basic schema definition :-

The syntax of create table command is

create table r
(A_1 D_1, A_2 D_2, ---- A_n D_n
<integrity - constraint$_1$ >,
<integrity - constraint$_2$ >

Scanned by CamScanner

choclate

<integrity constraint n>

);

Where $A_1, A_2 \ldots A_n$ are the attribute names

$D_1, D_2 \ldots D_n$ are the datatypes which is associated with attributes.

The integrity constraints are optional.

The integrity constraints are primary key, foreign key not null, unique.

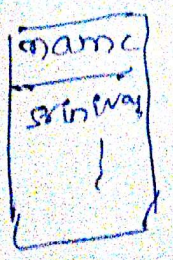
Create all the tables in the University database that means the table names are dept_name, course, instructor, section, teaches

Basic structure of SQL queries :-

Queries on single relation:

1. find the names of all instructor

select name from instructor;



2. find all the dept of all instructors

select dept_name from instructor;



Scanned by CamScanner

select distinct dept_name from instructor;

Dept_name
CSE (No duplicates)

3. find the names of all instructor in CSE dept who have salary > 70000.

select name from instructor where salary > 70000 and dept_name = CSE;

Queries on multiple relation :-

So far our example Queries were on a single relation. Queries often need to access information from multiple relations; for eg. Suppose we want to answer the Query retrive the names of all instructor along with their dept_names and dept_building name.

Select name, Instructor.dept_name, building from Instructor, dept. where instructor.dept_name = dept.dept_name;

name	dept_name	buildin
Srinivasan	Computerscience	SRK

Scanned by CamScanner

1. Find the teachers who teach some course.

Select name, course_id, ~~teachers~~ from instructor, teaches where Instructor.id = teaches.id;

Suppose we want to find Instructor name & course_id in the CSE department.

> Select name, course_id from instructor, teaches where Instructor.id = teachers.id and Instructor.dept_name = CSE;

Natural join:

Consider the Query for all the instructors in the University who have taught some course, find their names and course_id they taught.

> Select name, course_id from instructor, teachers where Instructor.id = teachers.id;

The Query can be written by using Natural join operation in SQL

> Select name, courseid from instructor natural join (⋈) teaches;

List the names of instructors along with their titles of courses that they teach.

Select name, title from instructor ⋈ teaches from, course where teaches.course_id = course.course_id

Scanned by CamScanner

Additional basic operations:-

The rename operation:

Consider again the Query that we used earlier

> select name, cource-id from instructor, teaches where instructor.id = teaches.id.

> select name as instructor-name, cource-id from instructor, teachers where instructor.id = teacher.id

Instructor_NAME	cource-id

For all instructors in the University who have tought some cources.

find the names of cource-id all the cources they tought.

select T.name, S.cource_ID from instructor as T, teaches as S where T.ID = S.ID;

String operations:

They are so many string functions supported by SQL. Upper(s), lower(s), Trim(s), Like(s)

1. percentage: (%)

The percentage character matches any substring.

Scanned by CamScanner

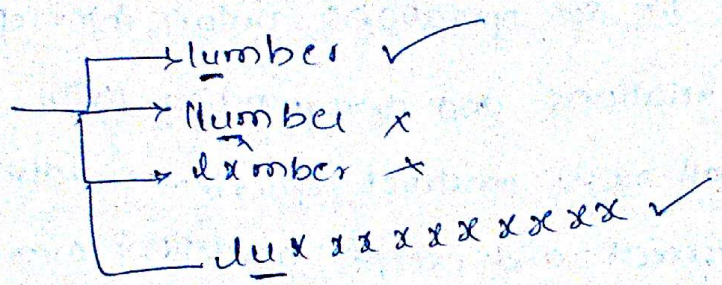
2. underscore (_) :-

The underscore character matches any character for Eg: '% comp %' matches any string containing as a substring. for Eg: 1. introduction to computer science.

2. computational biology

3. computer science

'_ u %'



Ordering the display of tuples:

- List alphabet order all instructors in the physics department:

Select name from instructor where dept-name = 'physics' order by name ; DESC
ASC

→ after orderby we use - DESC for descending ASCE for ascending. by default order by takes ascending order

where clause predicate:

~~Select~~

find the names of all instructors with the salary amount between 90000 and 100000.

Scanned by CamScanner

> select name from instructor where salary <= 100000 and salary >= 90000 ;

Another form

> select name from instructor where salary between 90000 and 100000.

Set operations:

The set operations union, interception, except operate on relations and corresponding to the mathematical. we shall now construct queries involving the union, intersect and except operations over 2 sets.

1. find the set of all courses taught in fall 2009 semester

> select courseid from section where semester='fall' and year = 2009.

2. find the set of all courses taught in the spring 2010 semester

> select courseid from section where semester='spring' and year = 2010

3. To find the set of all courses taught either in fall 2009 or in spring 2010, or both.

> select course-id from section where semester ='fall' and year=2009

union

select course-id from section where semester = 'spring' and year=2010.

Scanned by CamScanner

4. To find the set of all courses taught

The union operation automatically eliminates all the duplicates. if you need duplicates in the result use unionall.

The Intersect operation:

1. find the set of all courses start in the fall 2009 as well as in spring 2010.

>select course-id from section where semester='fall' and year=2009 intersect select course-id from section where semester='spring and year=2010

The except operation:

1. find all courses taught in fall 2009 semester but not in spring 2010.

>select course-id from from section where semester='fall' and year=2009 except select courseid from section where semester='spring' and year=2010

Aggregate functions:

Aggregate functions are functions that take a collection of values as input and return a single value.

SQL offers 5 built in aggregate functions.

1. average: avg()
2. minimum: min()
3. maximum: max
4. total: sum
5. count: count.

Scanned by CamScanner

Modification of the database:

* Delete: A delete requires arguments as a query

for eg: delete from r where p

where r is a relation and p is a condition.

> The delete is used for deleting the tuples (rows)

1. Delete all the tuples in the instructor relation pertaining to instructors in the finance dept

> delete from instructor where dept-name='finance'

2. Delete all instructors with a salary blw 13,000 and 45000

delete from instructor where salary <=45000 and salary >=13000.

Insertion:

Insert into student select * from stu;

Update

Eg: update instructor set salary = salary * 1.05;

Scanned by CamScanner